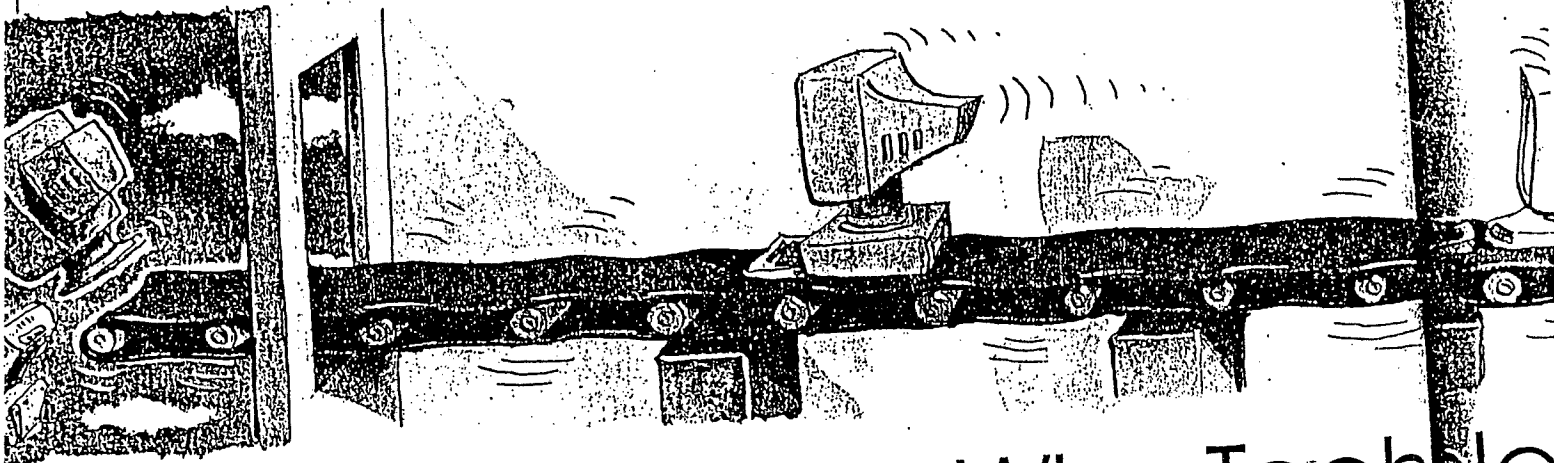


2 copies

Inventing – and reinventing – the proprietary architectures for open systems



How Architecture Wins Technology

by Charles R. Morris and Charles H. Ferguson

The global computer industry is undergoing radical transformation. IBM, the industry's flagship, is reeling from unaccustomed losses and is reducing staff by the tens of thousands. The very survival of DEC, the industry's number two company, is open to question. A roll call of the larger computer companies – Data General, Unisys, Bull, Olivetti, Siemens, Prime – reads like a waiting list in the emergency room.

What's more, the usual explanations for the industry's turmoil are at best inadequate. It is true, for example, that centralized computing is being replaced by desktop technology. But how to explain the recent troubles at Compaq, the desktop standard setter through much of the 1980s? Or the battering suffered by IBM's PC business and most of the rest of the desktop clone makers, Asian and Western alike?

And the Japanese, for once, are unconvincing as a culprit. The fear that Japanese manufacturing prowess would sweep away the Western computer industry has not materialized. True, Japanese companies dominate many commodity markets, but they have been losing share, even in products they were expected to control, like laptop computers. Earnings at their leading electronics and computer companies have been as inglorious as those of Western companies.

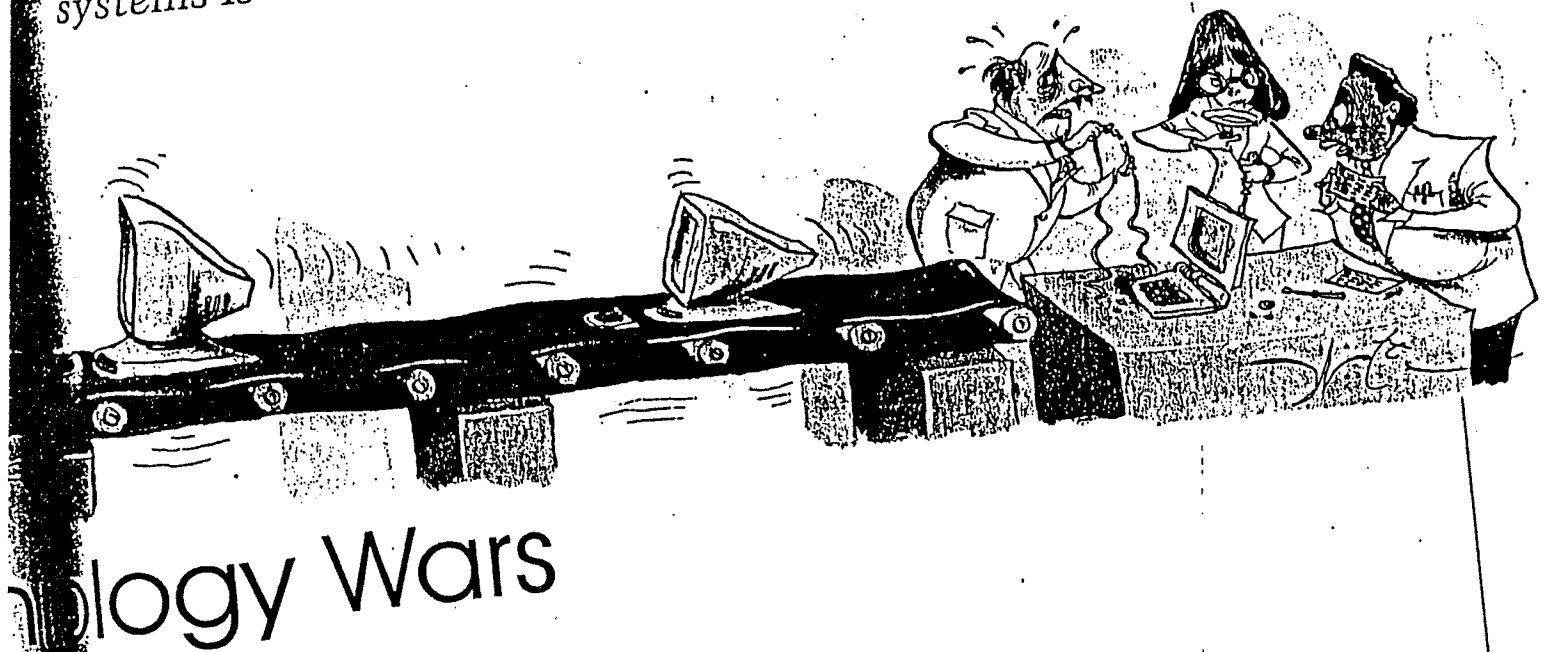
Explanations that look to the continuing shift in value added from hardware to software, while containing an important truth, are still too limited. Lotus has one of the largest installed customer bases in the industry. Nevertheless, the company has been suffering through some very rough times. Meanwhile, Borland continues to pile up losses.

Nor are innovation and design skills a surefire recipe for success. LSI Logic and Cypress Semiconductor are among the most innovative and well-managed companies in the industry, yet they still lose money. Design-based "fabless," "computerless" companies such as MIPS have fared very badly too. MIPS was saved from bankruptcy only by a friendly takeover. And Chips and Technologies is in dire straits.

Government protection and subsidies are no panacea either. The European computer industry is the most heavily subsidized in the world but still has no serious players in global computer markets.

Charles R. Morris is a partner in Devonshire Partners, a Cambridge, Massachusetts technology consulting and financial advisory firm. Charles H. Ferguson, an MIT Ph.D. and former MIT researcher, is an independent consultant, also in Cambridge. This article is based on their book Computer Wars: How the West Can Win in a Post IBM World, which was just published by Times Books.

systems is critical to competitive success.



Scale, friendly government policies, world-class manufacturing prowess, a strong position in desktop markets, excellent software, top design and innovative skills—none of these, it seems, is sufficient, either by itself or in combination with each other, to ensure competitive success in this field.

A new paradigm is required to explain patterns of competitive success and failure in information technology. Simply stated, competitive success flows to the company that manages to establish proprietary architectural control over a broad, fast-moving, competitive space.

Architectural strategies have become of paramount importance in information technology because of the astonishing rate of improvement in microprocessors and other semiconductor components. The performance/price ratio of cheap processors is roughly doubling every eighteen months or so, sweeping greater and greater expanses of the information industry within the reach of ever-smaller and less expensive machines. Since no single vendor can keep pace with the deluge of cheap, powerful, mass-produced components, customers insist on stitching together their own local system solutions. Architectures impose order on the system and make the interconnections possible.

An architectural controller is a company that controls one or more of the standards by which the

entire information package is assembled. Much current conventional wisdom argues that, in an "open-systems" era, proprietary architectural control is no longer possible, or even desirable. In fact, the exact opposite is true. In an open-systems era, architectural coherence becomes even more necessary. While any single product is apt to become quickly outdated, a well-designed and open-ended architecture can evolve along with critical technologies, providing a fixed point of stability for customers and serving as the platform for a radiating and long-lived product family.

Proprietary architectures in open systems are not only possible but also indispensable to competitive success—and are also in the best interest of the consumer. They will become increasingly critical as the worlds of computers, telecommunication, and consumer electronics continue to converge.

Architectures in Open Systems

In order to understand architecture as a tool for competitive success in information technology, consider first the many components that make up a typical information system and the types of companies that supply those components.

Take the computer configuration in a typical Wall Street trading or brokerage operation. Powerful workstations with 50 MIPS (millions of instructions per second)—comparable to the power of standard mainframes—sit on every desk. The workstations are connected in a network so they can communicate with each other or with several

Proprietary architectures are not only possible but also indispensable to competitive success.

others at a time. Teams of workstations can be harnessed together to crunch away on a truly big problem. Powerful computers called servers support the network and manage the huge databases—bond pricing histories, for instance—from which the workstations draw.

Such a modern network will be almost entirely open, or externally accessible by other vendors; critical elements, from perhaps as many as a hundred vendors, plug interchangeably into the network. The workstations themselves are from companies like Sun Microsystems, Hewlett-Packard, and IBM, or they may be powerful personal computers from Apple or any of a number of IBM-compatible PC manufacturers. IBM and Hewlett-Packard make their own workstation microprocessors; most workstation or personal computer makers buy microprocessors from companies like Intel, Motorola, Texas Instruments, LSI Logic, AMD, and Cyrix. Almost all the display screens are made in Japan by Sony, NEC, and many other companies; the disk drives come from American companies like Seagate or Conner Peripherals. The memory chips are made in Japan or Korea. The network printers will typically have laser printing engines from Japan or, if they are high-performance printers, from Xerox or IBM; the powerful processors needed to control modern printers will come from AMD, Motorola, or Intel. The rest of the standardized hardware components on the network, like modems, accelerator boards, coprocessors, network interface boards, and the like, will be made by a wide variety of Asian and American companies.

The network will have many layers of software, most of it "shrink-wrapped" from American companies. The operating system—the software that controls the basic interaction of a computer's components—may be a version of AT&T's UNIX, specially tailored by the workstation vendor, as with Sun and IBM, or it may come from a third party, like

Microsoft. Many vendors, like Lotus and Borland, will supply applications software. The complex software required to manage the interaction of the servers and workstations on the network will, in most cases, be supplied by Novell. The software that converts digital data into instructions for printer engines is sold by Hewlett-Packard, Adobe, or one of their many clones. Each smaller element in the system, like a modem or video accelerator, will have its own specialized software, often supplied by a vendor other than the manufacturer.

It is possible to construct open systems of this kind because for each layer of the network there are published standards and interface protocols that allow hardware and software products from many vendors to blend seamlessly into the network. The standards define how programs and commands will work and how data will move around the system—the communication protocols and formats that hardware components must adhere to, the rules for exchanging signals between applications software and the operating system, the processor's command structure, the allowable font descriptions for a printer, and so forth. We call this complex of standards and rules an "architecture."

A small handful of the companies supplying components to the network will define and control the system's critical architectures, each for a specific layer of the system. The architectural standard setters typically include the microprocessor designer (such as Sun or Intel); operating system vendors (possibly Sun or Microsoft); the network system (usually Novell); the printer page-description system (Adobe or Hewlett-Packard); and a small number of others, depending on the nature of the network. Each of these is a proprietary architecture; although the rules for transmitting signals to an Intel processor, for example, are published openly for all vendors, the underlying design of the processor

A small handful of innovative companies will define and control a network's critical architectures.

is owned by Intel, just as the design of Sun's operating system is owned by Sun, and so on for Microsoft's Windows/DOS, Novell's Netware, or Adobe's PostScript.

Companies that control proprietary architectural standards have an advantage over other vendors. Since they control the architecture, they are usually better positioned to develop products that maxi-

ze its capabilities, by modifying the architecture, they can discipline competing product vendors. In the open-systems era, the most consistently successful information technology companies will be the ones who manage to establish a proprietary architectural standard over a substantial competitive base and defend it against the assaults of both clones and rival architectural sponsors.

It has been conventional wisdom to argue that clones, and the cause of technological progress, are better served by *nonproprietary* systems architectures. This is emphatically untrue. There are many examples of nonproprietary architectures, like the CITT fax standard or the NTSC television standard, most of them established by government bodies or industry groups. Because they are set by committees, they usually settle on lowest-common-denominator, compromise solutions. And they are hard to change. The NTSC has been upgraded only once (for color) in a half-century; committees have been squabbling over an improved fax standard for years. *Proprietary* architectures, by contrast, because they are such extremely valuable franchises, are under constant competitive attack and must be vigorously defended. It is this dynamic that compels a very rapid pace of technological improvement.

Architectural Competitions

The computer industry has been competing on architecture for years. Take the example of the product that established IBM's dominance in the mainframe computer business—the IBM System/360. The 360 was arguably the first pervasive, partially open, information technology architecture. In the late 1960s, once the System/360 became the dominant mainframe solution, IBM began to unbundle component pricing and selectively open the system, in part because of government pressure. Published standards permitted competitors and component suppliers to produce a wide range of IBM-compatible products and programs that were interchangeable with, and sometimes superior to, IBM's own. By licensing its MVS operating system to Amdahl, for example, IBM made it possible for Fujitsu, Amdahl's partner, to produce clones of the IBM mainframe. Much of what was not licensed away voluntarily was acquired anyway by the Japanese through massive intellectual property theft.

Hundreds of new companies selling IBM-compatible mainframe products and software placed in-

tense competitive pressure on IBM. But they also assured that the IBM standard would always be pervasive throughout the mainframe computing world. As a result, even today IBM controls some two-thirds of the IBM-compatible mainframe market and an even higher share of its profits, not only for central processing units but also for disk drives, systems software, and aftermarket products like expanded memory. Because they have no choice but to maintain compatibility with the IBM standard, competitors must wait to reverse-engineer IBM products after they are introduced. Typically, by the time competitive products are on the market, IBM is well down the learning curve or already

I For over 20 years in the mainframe business, IBM has played this game brilliantly and won every time.

moving on to the next generation. And as the owner of the dominant architecture, IBM can subtly and precisely raise the hurdles whenever a particular competitor begins to pose a threat. For over 20 years, in generation after generation, IBM has played this game brilliantly and won every time.

Ironically, IBM badly fumbled an equivalent opportunity in desktop computing, handing over the two most critical PC architectural control points—the systems software and the microprocessor—to Microsoft and Intel. Since any clone maker could acquire the operating system software from Microsoft and the microprocessor from Intel, making PCs became a brutal commodity business. As a high-cost manufacturer, IBM now holds only about 15% of the market it created.

In a related error, Compaq made the mistake of assuming that IBM would always control the PC architectural standard. On that premise, the company geared its cost structure and pricing policy to IBM's, only to find itself almost fatally vulnerable when the savage PC price wars of the early 1990s exposed the commoditized character of PC manufacturing. Tellingly, while IBM and Compaq struggle to eke out profits from their PC businesses, Microsoft and Intel are enjoying after-tax margins of about 20%, on sales of more than \$4 billion and \$6 billion respectively, and together they have more cash than IBM.

For a similar example, consider the case of Lotus. Lotus got its start in a market—spreadsheet software—where products are complex and feature-rich, hardly commodities. And over the years, the

company acquired or developed a broad array of other products—Jazz, Manuscript, Improv, AmiPro, Notes, and Freelance—some of which are technically excellent. Lotus's competitive problem, however, is that these products lack any deep architectural commonality. Indeed, even the embedded spreadsheet software in the company's various offerings is incompatible from one to another.

Point product vendors like Lotus can be very profitable for a time. However, they are always at risk when an architectural leader changes the rules of the game. For example, while Lotus was accumulating a grab bag of point products, Microsoft was creating an architectural lock on the graphical user interface (GUI) for DOS-based computers. (See the insert "Scenarios for Architectural Competition: Graphical User Interfaces.") And Windows now defines the environment in which Lotus's software must compete. The great power of Windows is that it creates a relatively simple, intuitive, and reasonably uniform interface between a user and a very wide range of applications software. As users become accustomed to the greater ease of Windows, they insist on it, and point product vendors like Lotus are forced to adapt their software to run under the Windows architecture. But Microsoft also offers

its own line of point products, like Excel and Word, and since they arguably better exploit the Windows architecture, they are steadily encroaching on Lotus's market share.

The irony is that for a time in the 1980s, Lotus had such a powerful market position that it almost certainly could have established a GUI standard itself. But the company neglected to do so. Such strategic errors spell the difference between an architectural winner and loser.

Principles and Phases of Architectural Competition

There are five basic imperatives that drive most architectural contests:

1. Good products are not enough. Products distribute architectures and can contribute to the success of an architectural strategy. However, as the case of Lotus suggests, good products alone are not enough. But if the sponsor invests heavily in continuous product improvement, products of only modest capabilities can become the basis for architectural leadership. For example, both Zilog and

Scenarios for Architectural Competition: Graphical User Interfaces

Graphical user interfaces (GUIs) are the software that permits users to maneuver around applications visually—for example, issuing commands by pointing at icons—providing a simple, consistent method of working with many different programs. The evolution of the GUI market provides a dramatic example of the dynamics of architectural competition.

The original GUI was developed at Xerox's famed Palo Alto Research Center (PARC) and unveiled with the Xerox Star in the early 1980s. The Star was a brilliant achievement for its time—a high-performance, very expensive, easy-to-use networked workstation. But it was a completely closed system; there was no published applications-program interface, so no one but Xerox could supply software to run on the Star. Its appeal was therefore far too limited ever to become a pervasive desktop standard.

Steve Jobs adapted the Star technology to Apple, but it took several tries before Apple began to make inroads in the GUI arena. Apple's first try was the Lisa, a substantially closed system that failed to attract any market share. The company got it more nearly right with the Macintosh. At least in later incarnations, the Mac has been hospitable to third-party software devel-

opers. It is considerably less expensive than the Lisa and has a superb operating system/GUI architecture. But Apple has still sharply limited its distribution potential by insisting on bundling its architecture with only its own second-rate hardware. The Mac is hardly a failure, but had Apple licensed its systems software broadly, Apple and its microprocessor partner, Motorola, could have exercised the same architectural control over personal computing that Microsoft and Intel do now.

The operating system/GUI architectural struggle is far from over and will be one of the most heated competitive arenas of the 1990s. IBM OS/2 2.0 is technically excellent but suffers from a very late start. Microsoft's brand new NT system, which will run Windows applications, will raise the hurdles yet again. A variety of UNIX-based standards are alternatives to systems derived from the original DOS. And IBM and Apple have joined forces on a next-generation operating system/GUI in their Taligent partnership.

As the ongoing GUI contest suggests, architectural battles are fast-moving, hotly challenged, and rarely completely settled. The rewards to a winner, however, can be great.

Scenarios for Architectural Competition: Video Games

The home video game industry, dominated by Nintendo and Sega, is a serious industry. Some 30 million American homes, or about 70% of all homes with a child between the ages of eight and twelve, own a video game. Both Nintendo and Sega sell video game consoles (basic, 16-bit, 286-level computers) with tightly bundled operating systems. Game software is developed by independent vendors under tightly controlled licenses but distributed only through the two companies' networks at hefty markups. Profits flow from game sales, not consoles.

Bundled architectures are ripe for attack by more open systems, just as the Apple II was overwhelmed by the IBM PC. In fact, a number of American companies have targeted the game market. Electronic Arts, for one, has won a copyright suit allowing it to reverse engineer Sega's operating system. The availability of a Sega system clone would break that company's hold over game software and open up console manufacturing to clones. Another company, 3DO (formerly the San Mateo Software Group), has plans to release a powerful consumer-oriented operating system that will be ideally suited for games. Specifications have been provided to a number of Asian manufacturers, in-

cluding Matsushita, a 3DO investor, for a CD-ROM-based console. Existing best-selling games, presumably, could readily be adapted to the new system. 3DO's objective is to own a Windows-like architectural franchise in the consumer world.

An interesting and potentially formidable dark-horse competitor is Silicon Graphics, a company that has built its industry-leading three-dimensional image manipulation technology into a billion-dollar business. From its original base in the engineering/CAD industry, Silicon Graphics has found a new niche supplying the technology behind the spectacular special effects in Hollywood hit movies. Terminal for all, these systems could produce mind-boggling game effects. Silicon Graphics is known to have a consumer game strategy under way.

All these companies have ambitions that extend well beyond boys' games. In fact, the first of these lines of image-oriented consumer platforms for every line from new services to home shopping to endless entertainment services. On the principle that the low end always wins, such platforms eventually may upplant the current generation of personal computers. Microsoft and Intel beware.

AMD at various stages in the PC microprocessor contest made Intel-compatible chips that were superior to Intel's own, but neither company matched Intel's commitment to R&D, and both were left behind as Intel rolled out one generation of improved processor after another. Once an architecture is established, it in turn becomes a distribution channel for additional products, with the architectural controller's products holding the favored position.

2. Implementations matter. Manufacturing decisions are playing an increasingly important role in product strategy. But since successful architectures have a high design content and usually a high software content, manufacturing skills by themselves are not sufficient to prevail in architectural competition. Japanese and other Asian companies, for example, despite their great manufacturing prowess, have only rarely established architectural franchises. Generally, they have settled for positions as clone makers or commodity implementors. Perhaps the only area where Japanese companies have established proprietary control over an important architectural space is in video games. But even the leaders in this arena, Nintendo and Sega, are at risk. (See the insert "Scenarios for Architectural Competition: Video Games.")

While insufficient on their own, however, manufacturing skills may well be an essential competence for success in the architectural contests of the 1990s. The reason: implementation is increasingly becoming the key to winning architectural control. In microprocessors, for example, a good implementation can improve performance by a factor of two. That's why architectural leaders like Intel typically make their own chips. By contrast, Sun Microsystems has chosen to focus solely on the design of its

Manufacturing skills may well be essential for success in architectural contests.

SPARC microprocessor, a decision that has been a source of recent difficulty for the company because subpar supplier implementations have compromised SPARC performance. High-quality implementations are equally important in the new generations of hand-held computers. Indeed, the more advanced information technology makes inroads into consumer markets, the more manufacturing skills will prove invaluable.

3. Successful architectures are proprietary, but open. Closed architectures do not win broad franchises. Choosing the right degree of openness is one of the most subtle and difficult decisions in architectural contests. IBM opened its PC architecture too broadly—it should have, and could have, retained control of either or both the operating system and microprocessor standard. Apple made the opposite mistake of bundling the Mac operating system too closely to its own hardware. Sun, in contrast to Apple, opened its SPARC RISC architecture very early, both to software developers and processor cloners; it has the lead position in workstations, and its broad base of third-party software support has helped maintain customer loyalty through a series of technical stumbles. Autodesk's computer-aided design (CAD) software for builders is open to add-on third-party packages, like kitchen design tools, and its broad base of supporting software has given it control of a small but very profitable franchise.

4. General-purpose architectures absorb special-purpose solutions. Architectures that cannot evolve to occupy an ever-broader competitive space are dead ends. Wang's lucrative word processor franchise was absorbed by general-purpose PCs. Special-purpose CAD workstations from Daisy, Applicon, and others were absorbed by more general-purpose desktop machines. Special-purpose game machines will, in all likelihood, be absorbed by more general-purpose consumer systems.

5. Low-end systems swallow high-end systems. Minicomputers poached away huge chunks of mainframe territory and were assaulted in turn by workstations and networks. Workstations are under pressure by increasingly high performance PCs. Traditional supercomputers and very high-end mainframes are vulnerable to parallel arrays of inexpensive microprocessors. High-end data-storage systems are similarly under attack from arrays of inexpensive, redundant disks. Although IBM helped create the personal computer revolution, it steadfastly refused to recognize its implications. Until relatively recently, it even called its desktop products division "Entry Systems," ignoring the fact that today's microprocessor-based machines are a replacement for traditional computers, not an entry point or way station to them.

However, managers must keep in mind that even those companies that best follow these principles are not necessarily guaranteed contin-

ued success in the marketplace. Architectural contests typically move through a number of different phases, and only those companies that successfully

I Architectures that cannot evolve to occupy an ever-broader competitive space are dead ends.

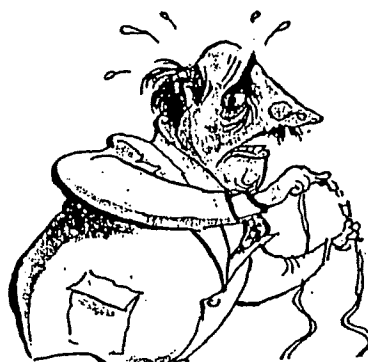
navigate them all, maintaining their pace and direction in the fluid environment of rapidly evolving technologies, emerge as winners over the long term. It's a delicate balancing act, and one that requires ever-increasing flexibility as the technologies mature.

There are five principal phases to architectural competition:

Commitment. Architectural challenges usually emerge from the early-stage chaos of competing point products. Before the IBM PC, personal computers were rigid, closed systems that tended to bundle their own operating systems and applications software. Compaq had the insight that by purchasing a Microsoft operating system identical to that of the PC, it could ride the wave of the PC's success. Microsoft then insisted that all subsequent clone makers buy the same operating system and so seized the critical PC software architectural standard. Microsoft's insight was to realize that it was in an architectural contest and to take the appropriate steps, including steadily expanding the generality and scope of its systems to come out the winner.

Diffusion. Large profits come from broad franchises. Open architectures are successful because they can be broadly diffused. Xerox's Interpress page-description software, which converts digital data into printer instructions, is excellent but can be purchased only with Xerox high-end printers. Adobe, by contrast, has widely licensed its PostScript language and has become the industry standard setter. Intel widely licensed the early versions of its xx86 processors, then sharply restricted licensing of its 386 chip after the Intel standard had become firmly entrenched. IBM, on the other hand, has long resisted diffusing its mainframe and minicomputer software.

Of course, diffusion decisions are not without risk. Once again, balance and timing are essential. For example, Philips licensed its com-



crease market penetration. But Sony outperformed Philips and took half the market. Philips's standard was a static one that it never developed further.

Lock-in. A company has a "lock" on an architecture when competitors are trained to wait until the architectural leader introduces each new product generation. Intel and Microsoft, at least temporarily, seem to have achieved this position in PC markets. Sun was on the verge of a locked-in franchise in workstations but may have fallen short; the performance of its SPARC RISC processor design has been lagging behind the competition, and the company neglected to solidify its franchise by moving rapidly down to lower end platforms.

But lock-in is sustainable only when a company aggressively and continuously cannibalizes its own product line and continually and compatibly extends the architecture itself. This is a strategic choice that many companies find difficult to make. Often, managers become overprotective of the products that brought them their original success. IBM, for example, has frittered away a powerful lock on back-office transaction processing and operating systems. In a misguided effort to protect hardware sales, it has refused to release products, long since developed internally, that would adapt its best-selling AS400 minicomputer software to the RS6000 workstation. Such reflexive self-protection simply hands over a valuable franchise to the Microsofts and other vendors storming up from the low end.

Harvest. Of course, the ultimate objective of architectural competition is to win a market leader's share of the profits. Just to give one dramatic example, profit margins on Intel's *xx86* family of chips are in the 40% to 50% range and account for well



Just as products must be cannibalized, so must architectures themselves. The better the architecture, the longer its lifespan; but sooner or later every architecture, no matter how well designed, becomes obsolete. And before it does, the market leader must be prepared to move ahead, to do away with the old and introduce the new. Industry leaders often fail to cannibalize their old architectures, but al-

though nothing is more painful, to do so is absolutely necessary. Otherwise, competitors quickly move to create and introduce rival franchises, and these eventually dominate the industry. IBM's failure to cannibalize its mainframe and minicomputer franchises provides a stark example of the catastrophic effects of waiting too long.

DEC provides another example. The company developed outstanding RISC products very early. But DEC declined to cannibalize its profitable VAX-VMS architecture because its VMS operating system, the source of its franchise, was tightly integrated with its aging VAX hardware. Predictably, DEC was beaten out by vendors such as Sun Microsystems and Microsoft, which didn't hesitate to move in with their newer, more powerful alternatives. (The main developer of DEC's advanced systems, Dave Cutler, is now in charge of developing NT for Microsoft.)

There are three lessons here. First, with better architecture DEC could have kept VMS alive longer. If VMS had been "portable," that is, not restricted to VAX hardware, DEC could have ported VMS to other vendors' hardware, making VMS an industry standard. Indeed, the company could have used RISC technology itself without losing its VMS franchise. Second, DEC would have been better off cannibalizing itself, rather than waiting to be cannibalized by others.

The third lesson, though, is the most important. As DEC's experiences with VMS and IBM's mistakes with the mainframe and minicomputer franchises show, the cultural and organizational structures useful for managing traditional, closed, integrated businesses will not work for companies that intend to compete with architectural strategy. In fact, we believe that architectural competition is stimulating the development of a new form of business organization.

This new structure, which we call the Silicon Valley Model, has major implications both for information technology and for many other indus-

I Though painful, it is absolutely necessary to cannibalize old architectures.

over 100% of the company's earnings. But no locked-in position is ever completely safe, and companies must be careful when they harvest not to rest on their previous successes. Indeed, Intel may have harvested too aggressively, drawing out spirited recent attacks by clone makers such as AMD and Cyrix.

Scenarios for Architectural Competition: Page- and Image-Description Standards

Page- and image-description standards are rapidly evolving from their initial base in printers into a very large business that will transform the entire printing and publishing industry. Probably most published material is now captured in electronic format, and a major competition is shaping up for control of the standard for storage, transmission, and manipulation of complex text, images, and multimedia documents. The technology involved is extraordinarily sophisticated and processing-intensive. Data compression and decompression and image-manipulation algorithms tax all but the very fastest of available processors; data storage requirements are very large; and requirements for communications capacity outpace most conventional systems. All these hurdles are falling very rapidly before a wide range of technical advances.

At the moment, Adobe must be considered the front-runner in the standards contest. Its Acrobat product, due to be introduced this year, will provide the industry's most advanced storage, compression, and transmission capabilities. The first versions will permit users to annotate, but not edit, electronically stored texts. Later releases are expected to include editing options. Microsoft is mounting a major challenge; at least in the word processing of documents and fonts. The dark horse is Xerox, which traditionally has possessed a vast array of image- and text-oriented technologies that it somehow never manages to commercialize. A number of smaller companies have also planted their pennants, including, refreshingly, two from Europe: Harlequin and Hyphen. Hewlett-Packard and Microsoft have formed an alliance to stay in contention, but their solutions are, for the moment at least, quite limited.

An early inning in the contest will involve the possibility of creating a new proprietary fax standard. The combination of faxes with high-quality plain-paper printers could induce a very substantial increase in fax usage, particularly if images are of sufficiently high quality to transmit pictures, working drawings, and the like. Two new products, PostScript for Fax from Adobe and Satisfaction from Intel, provide much-improved resolution and decrease the required data compression to allow existing low-capacity communication systems to handle complex images. Both interconnected with standard fax machines to send and receive low-resolution images.

tries. The model is still young and rapidly changing, and although Microsoft probably comes closest, no company fits it perfectly.

Managing Architectural Competition: The Silicon Valley Model

The Silicon Valley Model arose a decade ago when early architectural competitors noticed that they faced the same problems in managing organizations that they faced with technologies and architectural strategies.

In retrospect, this is not surprising. Architecture responds to the same imperatives in both systems and organizations. It reduces complexity. It permits clean separation between centralized general-purpose functions and decentralized or specialized functions. It enables management of unpredictability and change; individual technologies, components, or products can be switched without the need to redo everything. For similar reasons, good architecture facilitates experimentation and competition: once the framework is specified, multiple approaches can compete without jeopardizing compatibility. And finally, a standard architecture permits many systems and organizations to be developed independently and still work together gracefully.

As an organizational paradigm, the Silicon Valley Model therefore has several characteristic features and advantages. Following are the most important:

1. Organizational architecture and decision making that mirror technical architecture. Any organization should develop and use good technical architectures. But Silicon Valley Model firms take an additional step: the structure of the firm itself mirrors the technical architectures it uses.

Thus, for example, Microsoft is structured so that its existing systems software and applications software are managed separately, as are new architectural efforts such as NT. In this manner, Microsoft can diffuse its applications across multiple operating systems (both its own and others, like the Apple Macintosh), while also marketing its operating systems by courting other vendors' applications. The two businesses can work largely independently, yet only Microsoft gains the benefits of their synergism. Most decisions can be made directly within the organization responsible for the relevant architectural domain; this minimizes complex vertical and horizontal debates.

2. Meritocracy and direct feedback. Silicon Valley Model firms enable and force direct performance

feedback, at levels ranging from individuals to business units. At Microsoft, team members rate each other periodically in peer reviews. Outstanding performers are rewarded; laggards are warned, then fired. Technical expertise is required for a large fraction of senior management, and communication occurs directly between the relevant parties, unbuffered by hierarchy.

By contrast, performance ratings in traditional bureaucracies are determined by managers at higher levels, and compensation is rarely based on long-term corporate performance. The process is often heavily politicized; dissent is suppressed, and incompetence goes unpunished.

Architectural competition also exposes Silicon Valley Model firms to another form of peer review—product competition. To succeed as industry standard setters, firms must license their architectures to competitors, while also developing critical products themselves. As a result, each layer of the firm (and of the architecture) is exposed to direct competition and market feedback. Hence although Microsoft controls Windows, application groups still compete individually: Excel against Lotus and QuattroPro, Word against WordPerfect and AmiPro, and so forth. Architectural leadership provides an advantage, but prevents a cover-up. Silicon Valley Model firms are structured so that excellence is the only defense.

3. Clean boundaries, both internal and external. In architected corporate structures, organizations can create and dissolve alliances rapidly, both internally and externally. Organizations are very flat, and development groups have simple, clean interfaces to each other determined by architectural boundaries. Architecture and point products can be

Silicon Valley Model firms take an additional step: the structure of the firm itself mirrors the technical architectures it uses.

kept apart. Moreover, products can invisibly incorporate architected "engines" developed by other organizations, including competitors. For example, a start-up called InfoNow has organized alliances involving itself, Microsoft, publishers, computer vendors, and other software companies. InfoNow packages software products, together with reviews and samples of them, which are preloaded for free on computers; the software products, however, are en-

crypted. Users can sample them, read reviews, and then purchase them by telephone, which triggers electronic decryption. Adding new software packages is trivial.

4. Internal proprietary control of architecture and critical implementations, externalized commodities and niches. Silicon Valley Model firms seek to externalize the maximum possible fraction of their total system, while carefully controlling those areas required to establish and hold an architectural franchise. Thus core development of the general purpose architecture is always internally controlled. So usually are critical product implementations, which cover the broadest markets and are required either for early diffusion or later harvesting.

Broad, cost-sensitive markets are the strategic high ground, if covered by proprietary architectures.

Silicon Valley firms also carefully manage their dependencies, so as not to become unilaterally dependent on architectural competitors.

On balance, however, Silicon Valley Model firms are much less autarkic than traditional large firms. Niche products, commodity components, and architectures controlled by others are outsourced, and/or relegated to licensees. In fact, Silicon Valley firms actively seek to commoditize regions not under their control.

This yields several benefits. For one, companies can focus on what they do best and on the efforts critical to architectural success. For another, broad outsourcing and licensing create competition among suppliers and licensees, which broadens the market and benefits the architectural leader. PC price wars delight Intel, Microsoft, and Novell; IBM and Compaq take the heat.

Interestingly, this contradicts the 1980s conventional wisdom that firms should avoid broad, cost-sensitive markets in favor of high-price niches. In fact, the broad market is the strategic high ground, if it is covered by a proprietary architecture. Niche product vendors can make profits, but they will remain minor players.

5. Migration and evolution over time. Just as architectures evolve and eventually become obsolete, so too with organizations. Thus the firm's internal structure and external alliances evolve along with its architecture and market position. As new layers are added to an existing architectural position (Windows on top of DOS, then NT underneath Win-

dows), new organizations are created; a similar situation occurs when an architecture must be cannibalized. Some Silicon Valley Model firms will soon face cannibalization; it will be interesting to see how they do.

Broader Implications of the Silicon Valley Model


The Silicon Valley Model is very much a product of a few companies in the computer sector, just as mass production was invented by Ford and just-in-time production by Toyota. And as in those cases, we believe that the Silicon Valley Model will diffuse throughout the broader information technology sector as the computer, telecommunications, information services, and consumer electronics industries merge.

In addition, however, as industrial competition in all industries becomes more complex and technological change accelerates, the model may have important effects upon many other fields. We think that it provides a framework that allows proprietary leaders in general to have the greatest span of control and profitability with the least complexity and smallest size. In fact, we think that the model is appropriate for small and large companies alike; it does, however, penalize unnecessary size. (Microsoft, with fewer than 15,000 employees, has a market capitalization equal to IBM's.) We will therefore close with an example of how architectural strategy and the Silicon Valley Model could have been used more than a decade ago, by Xerox.

Xerox became a large, global company through a single proprietary technology—xerography. Xerographic “marking engines” are the core of photocopiers, printers, and facsimile machines, all of which Xerox invented. But Xerox chose to exploit its control of xerography using the traditional strategy of integrated companies.

Where Xerox felt it could not develop products profitably itself, it simply left the market vacant. As a result, when the company's patent position eroded, Japanese competitors took the bulk of the blossoming low-end markets for personal copiers, laser printers, and fax machines. Xerox's market share declined from nearly 100% to about 30%.

Instead, Xerox could have developed an architecture for a broad family of machines and control systems, including interfaces for scanners, document handlers, and “finishers” for collating, stapling, and binding. It could have licensed its technology to other firms, and/or sold them xerographic engines. It could have developed products for core markets, leaving others to niche companies.

Every few years, the company could have changed or enhanced its architectures to improve its products and competitive position. The result could have been a Microsoft-like position, with Xerox holding the lion's share of the profits in a highly competitive, dynamic market—yet one under its own effective control. We think that similar strategies are available to companies in other complex industries— aerospace and machine tools, among others. If so, the information sector's strategic and organizational innovations might prove as interesting as its technology. 

Reprint 93203